LA-UR-18-22738
Approved for public release; distribution is unlimited.

Title: General Improvements to the MCNP Alpha-Eigenvalue Solver

Author(s): Josey, Colin James

Intended for: Report

Issued: 2018-03-29

General Improvements to the MCNP Alpha-Eigenvalue Solver

Colin Josey

March 26, 2018

Abstract

In this document, a new algorithm for computing the α-eigenvalue is implemented and tested in MCNP. The algorithm follows the traditional k–α method in which neutrons are followed from birth until fission in batches. During this simulation, a number of tallies are performed which are derived from integrating the transport equation over all phase space. These new tallies along with the collapsed transport equation are used to compute a physically valid α with or without delayed neutrons. The performance and accuracy of the method is then tested, in which it is found that the new algorithm typically outperforms the current implementation while also representing a more complete physical picture. Through a convergence analysis, it was found that many simple test problems had unexpectedly large biases for reasonable quantities of neutrons per batch. This occurred with both the old and the new algorithm. However, the tally method had lower biases for all problems tested. Special implementation considerations are also discussed.

1 Introduction

The α-eigenvalue is one of the possible solutions to the transient neutron transport equation. Via a separation of variables, the neutron flux is decomposed into a constant spatial component and an exponential time component. The coefficient in the exponent, α, then quantifies the asymptotic time behavior of the nuclear assembly. The resulting equation is similar in structure

to a k-eigenvalue equation. As such, the most common approach is the so-called k–α iteration method [3], in which α (which scales terms that are effectively sources or absorbers) is adjusted until k is equal to one.

The updating scheme for α is usually performed through a proportionality to $k-1$. This has a few drawbacks, namely that the proportionality must be carefully chosen, and that since k is insensitive to the difference between prompt and delayed neutrons, it tends to yield nonphysical solutions when delayed neutrons are considered. This work presents an alternative updater that is not directly a function of k and includes delayed neutrons directly. This new tally updater is implemented in MCNP and compared against the algorithm currently implemented.

In order to test this new implementation, a wide variety of tests were performed. To verify the accuracy, a deterministic infinite medium benchmark was created. To quantify the improvement in performance, delayed critical and prompt supercritical assemblies were run in direct comparison to the current k–α iteration. In addition, several tests examined the statistical distribution of α and the bias caused by statistics.

2 Theory

The approach to derive the α-eigenvalue equation starts by performing a separation of variables. Through the use of a Laplace transform, the neutron flux can be expressed in the form of Eq. (1) [2]. It should be noted that this form is not proven to be complete, but works sufficiently well for the following analysis.

$$\psi(\mathbf{r}, E, \hat{\mathbf{\Omega}}, t) = \sum_j e^{\alpha_j t} \psi_j(\mathbf{r}, E, \hat{\mathbf{\Omega}}) \tag{1}$$

If one sorts the αs such that $\Re\alpha_n > \Re\alpha_{n+1}$, then the asymptotic time behavior takes the form of Eq. (2). The same can be done with the delayed neutron precursors.

$$\lim_{t\to\infty} \frac{\partial \psi}{\partial t} = \alpha_0 \psi \tag{2}$$

The resulting value α_0 is the asymptotic logarithmic time derivative of the neutron flux. As such, it can characterize the maximum growth rate of

flux in a fissile geometry. In order to solve for α_0, the flux and precursor density is asserted to take the form of Eq. (3), where N is the number of precursor groups.

$$\begin{aligned} \psi(\mathbf{r}, E, \hat{\mathbf{\Omega}}, t) &= \psi(\mathbf{r}, E, \hat{\mathbf{\Omega}}) e^{\alpha t} \\ C_i(\mathbf{r}, t) &= C_i(\mathbf{r}) e^{\alpha t} \qquad i = 1 \text{ to } N \end{aligned} \tag{3}$$

This can be substituted into the neutron transport equation to take the form of Eq. (4). This equation follows the standard syntax of Bell and Glasstone, with the exception of the explicitly stated ν_s term, which corresponds to possible multiplicities through (n, 2n) reactions and similar. Also note that since $e^{\alpha t}$ is everywhere positive (assuming α is real, which is the case for the eigenvalue of interest), it can be divided out.

$$\begin{aligned} &\left(\frac{\alpha}{v(E)} + \hat{\mathbf{\Omega}} \cdot \nabla + \Sigma_t(\mathbf{r}, E) \right) \psi(\mathbf{r}, E, \hat{\mathbf{\Omega}}) \\ &= \frac{\chi_p(E)}{4\pi} \int_0^\infty dE' \nu_p(E') \Sigma_f(\mathbf{r}, E') \phi(\mathbf{r}, E') + \sum_{i=1}^{N} \frac{\chi_{di}(E)}{4\pi} \lambda_i C_i(\mathbf{r}) \\ &+ \int_{4\pi} d\Omega' \int_0^\infty dE' \nu_s(E') \Sigma_s(\mathbf{r}, E' \to E, \hat{\mathbf{\Omega}}' \to \hat{\mathbf{\Omega}}) \psi(\mathbf{r}, E', \hat{\mathbf{\Omega}}') \\ &\alpha C_i(\mathbf{r}) = \int_0^\infty dE' \nu_{di}(E') \Sigma_f(\mathbf{r}, E') \phi(\mathbf{r}, E') - \lambda_i C_i(\mathbf{r}) \end{aligned} \tag{4}$$

C_i can also be solved out, yielding Eq. (5).

$$\begin{aligned} &\left(\frac{\alpha}{v(E)} + \hat{\mathbf{\Omega}} \cdot \nabla + \Sigma_t(\mathbf{r}, E) \right) \psi(\mathbf{r}, E, \hat{\mathbf{\Omega}}) \\ &= \frac{\chi_p(E)}{4\pi} \int_0^\infty dE' \nu_p(E') \Sigma_f(\mathbf{r}, E') \phi(\mathbf{r}, E') \\ &+ \sum_{i=1}^{N} \frac{\chi_{di}(E)}{4\pi} \frac{\lambda_i}{\alpha + \lambda_i} \int_0^\infty dE' \nu_{di}(E') \Sigma_f(\mathbf{r}, E') \phi(\mathbf{r}, E') \\ &+ \int_{4\pi} d\Omega' \int_0^\infty dE' \nu_s(E') \Sigma_s(\mathbf{r}, E' \to E, \hat{\mathbf{\Omega}}' \to \hat{\mathbf{\Omega}}) \psi(\mathbf{r}, E', \hat{\mathbf{\Omega}}') \end{aligned} \tag{5}$$

This particular form has two differences from the typical transport equation. The first is the introduction of the α/v term on the left-hand side. When α is positive, this can be interpreted as an absorption cross-section. When α

is negative, this can be interpreted as a source. There are several different approaches to handling this source. Currently, MCNP adds $2|\alpha|/v(E)$ to both sides of the transport equation. The result is an (n, 2n) reaction with macroscopic cross-section $|\alpha|/v(E)$. Both resulting particles are identical to the incoming particle.

The second modification is the scaling of the delayed neutron ν by the factor $\lambda_i/(\alpha + \lambda_i)$. As the α eigenvalue approaches $-\min \lambda_i$, the population becomes dominated by delayed neutrons. The two necessary modifications (the scaling of ν and the modification to the sampling of χ_{di}) are not implemented in the most recent version of MCNP but have been added in in this development branch. It is worth noting that the singularity in this term guarantees that $\alpha > -\min \lambda_i$.

3 Current Methods

The k–α iteration technique takes Eq. (5) and re-inserts the value of k to scale both the delayed and prompt fission source. For each batch, α is treated as if it were static, scaling the ν_d components and providing either a source or absorber to the problem. Upon the completion of the batch, k is computed, and then α is updated such that $k \to 1$.

This stochastic root finding approach can have a number of issues. The most prominent one is that if the updating scheme $\alpha_{n+1} = f(\alpha_n, k_n)$ does not strictly enforce $\alpha > -\min \lambda_i$, nonphysical answers can occur. In addition, some choices for f can result in chaotic (and thus non-useful) solutions [9].

For MCNP, the current function f is given by:

$$\alpha_{n+1} = \alpha_n + (k_n - k_{\text{target}}) \left(\alpha_n + \frac{k_n}{\Lambda} \right)$$

where Λ is the mean neutron lifetime. k is chosen from either the collision, absorption, or track length estimator. k_{target} is the target k-eigenvalue, and is often set to one.

The algorithm used in MCNP does not update α at the end of every generation. Instead, each batch contains between two and four generations all simulated with the same α to allow for a more converged source. k and Λ are only computed in the last of these generations. While this technique can reduce the source dependency on previous step αs, and as a consequence reduce autocorrelation, a great deal of information is lost. Testing in Sec. 6.2

indicates that this approach reduces the figure of merit. As such, the new algorithm described in the next section only runs one generation per batch.

4 α Tally Method

This scheme makes only one modification to the current MCNP algorithm, which is to replace $f(\alpha_n, k_n)$ with a function of tallies. The gist of the idea is to take Eq. (5) and integrate it over all phase space. Under the assumption that the current guess of the flux is correct, Eq. (6) is then exact.

$$\begin{aligned}
&\alpha \int_0^\infty dE \int_V dV \frac{1}{v(E)} \phi(\mathbf{r}, E) + \int_0^\infty dE \int_V dV \Sigma_t(\mathbf{r}, E)\phi(\mathbf{r}, E) \qquad (6)\\
&\quad + \int_0^\infty dE \int_{4\pi} d\Omega \oint_S dS \psi(\mathbf{r}, E, \hat{\mathbf{\Omega}})\hat{\mathbf{\Omega}} \cdot \mathbf{n}\\
&\quad = \int_V dV \int_0^\infty dE' \nu_p(E')\Sigma_f(\mathbf{r}, E')\phi(\mathbf{r}, E')\\
&\quad + \sum_{i=1}^N \frac{\lambda_i}{\alpha + \lambda_i} \int_V dV \int_0^\infty dE' \nu_{di}(E')\Sigma_f(\mathbf{r}, E')\phi(\mathbf{r}, E')\\
&\quad + \int_V dV \int_0^\infty dE' \nu_s(E')\Sigma_s(\mathbf{r}, E')\phi(\mathbf{r}, E')
\end{aligned}$$

Every integral above can be tallied using Monte Carlo. Using the notation that $[\cdot]$ is a surface tally and $\langle \cdot \rangle$ is a flux-integrated volume tally simplifies the syntax to Eq. (7).

$$\alpha \left\langle \frac{1}{v} \right\rangle + \langle \Sigma_t \rangle + [\text{escape}] = \langle \nu_p \Sigma_f \rangle + \sum_{i=1}^N \frac{\lambda_i}{\alpha + \lambda_i} \langle \nu_{di} \Sigma_f \rangle + \langle \nu_s \Sigma_s \rangle \qquad (7)$$

In this form, it is clear that α can simply be solved for. The one root of interest is going to be a real value in the span $[-\min \lambda_i, \infty]$. Using a suitably large maximum value, this root can be found with bisection.

One can also make tallies both with track length and collision estimators. This results in two αs, α_{tr} and α_c. The track length α is used as the simulation α for the next batch. The choice of the track length estimator over the collision estimator or some linear combination of the two was arbitrary. Early testing indicated that the end result was fairly insensitive to this choice. Once the variance and covariance of each α estimator is known, the two can

be combined stochastically using the technique described in [13] to improve results. The final reported α is given by this combined estimate.

There are a number of technical details that need consideration in order to ensure the performance and quality of this algorithm. These details will be described in the next few sections.

4.1 Bias in α

The primary drawback of this technique is that, while the tallies that compose α are themselves essentially normally distributed via the central limit theorem, the combination of these values to compute α is not. In the prompt case, this effect is fairly minor; the function converting tallies to α is smooth and uncomplicated. This hypothesis is confirmed in Sec. 6.3.

In the delayed neutron case, however, the bias effect can be quite extreme. A simple example is a precisely delay-critical geometry. It is expected that α is precisely zero. However, even a slight increase in the prompt neutron production tally (which is effectively normally distributed) could push the geometry to prompt supercritical (with a corresponding very large α). Conversely, a slight decrease will not have a balancing effect, as α is bounded on the negative side. As such, α will be biased positive. For this reason, the skew and excess kurtosis of the distribution of α as well as the median are all reported at the end of the simulation (when using tally α), in order to indicate possible bias. This phenomenon is further investigated in Sec. 6.3.

4.2 Minimizing Cost of Tallying

As shown in Eq. (7), there are six kinds of tallies that need to be performed in order to calculate α. Two components require data that is not normally computed during a k-eigenvalue calculation and has not had access optimized.

For $\langle \nu_{di} \Sigma_f \rangle$, the function `acenu` has been substituted with `acenu_alpha`. If the simulation is not running an `ACODE` calculation, then it simply returns `acenu`. However, if `ACODE` is running, it computes a number of values which are stored in the new variable `rtc_alpha`. In order to accelerate transport, the α scaled ν_{total}, ν_d and p_i (the probability of a delayed precursor in group i) are calculated. These are then used to compute how many fission neutrons to bank as well as which χ spectrum to sample. To accelerate tallying, the α *un*scaled values for all three are also computed and stored.

With regards to $\langle \nu_s \Sigma_s \rangle$, things are a bit complicated. $\nu_s \sigma_s$ cannot be computed ahead of time due to $S(\alpha, \beta)$ effects. While σ_s is calculated implicitly during transport, it is not disambiguated such that computing ν_s is easy. As such, the residual of Eq. (7) is recast as Eq. (8).

$$r = \alpha - \frac{\langle (\nu_p - 1)\Sigma_f \rangle + \langle (\nu_s - 1)\Sigma_s \rangle - \langle \Sigma_c \rangle - [\text{escape}]}{\left\langle \frac{1}{v} \right\rangle} - \sum_i \frac{\lambda_i}{\alpha + \lambda_i} \frac{\langle \nu_{di} \Sigma_f \rangle}{\left\langle \frac{1}{v} \right\rangle} \tag{8}$$

Σ_c is already computed explicitly. $(\nu_s - 1)\sigma_s$ has a simple form in that it essentially only contains (n, xn) reactions. These occur at high energies, so the cross-section itself requires relatively little memory. This cross section is stored in `XSS` at pointer `JXS(28)`.

4.3 Tally Numerical Stability

It is important to note that the implementation of the above algorithm is very sensitive to implementation details. For example, when summed in parallel or in serial, the tallies can be slightly different. The relative error is often less than 10^{-12}, but it can be large enough to change the course of a single neutron. Once this occurs, a cascade effect will result in the final answer being completely different.

With that in mind, care was taken to ensure the numerical stability of the algorithm. An easy approach is to expand the precision of the numbers used for tallying. Quadruple precision was considered, but it slowed down simulation significantly. Instead, double-double arithmetic, using the techniques in [7], was implemented. All operations, including the individual tallies and the inter-thread summation use this expanded precision. When passed to the α computation, the heads and tails of the double-double number are added together and stored as a double. This significant expansion of numerical precision effectively eliminates the possibility of discrepancies between serial and parallel runs at minimal performance cost.

It is worth mentioning that the algorithms listed in [7] are sensitive to over-zealous optimization. For these algorithms, the precise order of operations is critical. Some compilers, such as the Intel compiler (version 17.0.1 tested), will perform algebraic transformations that are technically invalid due to the non-associative nature of floating point arithmetic. In order to

ensure the correct answer, some compiler flags may be necessary. For Intel, the flag `-fp-model source` was added during testing. Not adding this flag will, at worst, revert accuracy to double precision. Other compilers tested did not exhibit this effect using the default MCNP flags.

4.4 Statistical Numerical Stability

While the vast majority of MCNP uses the "square of sums, sum of squares" approach to computing the variance and covariance, this is not particularly numerically stable. When testing problems with small variances, such as the infinite medium test problems described in Sec. 6.1, this can result in inaccurate or even negative variances. To ensure stability, the mean and the variance were computed using Welford's method [14]. The covariance, skew, and excess kurtosis were calculated using Pébay's methods [10, 11].

As an aside, there are multiple formulas for the calculation of skew and excess kurtosis. In this particular implementation, skew is given by the G_1 estimator and excess kurtosis is given by the G_2 estimator in Eq. (9).

$$\begin{aligned} m_j &= \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^j \\ G_1 &= \frac{\sqrt{n(n-1)}}{n-2}\frac{m_3}{m_2^{3/2}} \\ G_2 &= \frac{n-1}{(n-2)(n-3)}\left((n+1)\frac{m_4}{m_2^2} - 3(n-1)\right) \end{aligned} \tag{9}$$

4.5 Avoiding Crashes with Negative α

Both the current version of MCNP and the new algorithm perform fission source iteration as the outer loop during the computation of α. There is a general issue in α-eigenvalue calculations when performed using fission source iteration. When the α time source is added to the problem, it is possible that the geometry can be "supercritical" on this source alone. As a simple example, consider the fictitious (n, 2n) reaction often used to handle the time source. The probability of doubling the weight of the particle is given by

$$P_{\text{double}} = \frac{\frac{|\alpha|}{v}}{\Sigma_t + \frac{|\alpha|}{v}}$$

As this value approaches 0.5, the expected total weight of the particle grows. Once the value exceeds 0.5, then the weight can grow without bound. In MCNP, the number of particles added to the fission bank per fission is proportional to the weight of the particle that caused the fission. If such a particle had repeatedly undergone (n, 2n) reactions, at best this will heavily bias the solution. At worst, the number of particles will exceed the fission bank, resulting in a crash.

This issue has lead to quite a bit of discussion in the literature. One modification replaces the (n, 2n) reaction, $\Sigma_\alpha = |\alpha|/v$ with an $\left(\mathrm{n}, \frac{1+\eta}{\eta}\mathrm{n}\right)$ reaction, $\Sigma_\alpha = \eta|\alpha|/v$ [16]. Adjusting the value of η towards zero increases the variance, which can help terminate chains earlier.

A second alternative is to cease iterating on the fission source and instead iterate on the α time source [12]. This also allows for a direct computation of α. Additionally, each batch forces the normalization of the time source. The result is an algorithm that is only valid for negative α, but otherwise always stable.

A third option is to approximate the transport equation. For example, the cross section $|\alpha|/v$ could be limited to some fraction of Σ_t or particles with weights higher than a threshold could be terminated. Both options would bias the simulation.

The problems tested in Sec. 6 have relatively small values of negative α. Even for the prompt α comparisons, extreme growth of particle weight was rare. As such, a weight cutoff of 100 was used in both the old and the new simulations. This allowed a direct comparison between the two without resorting to substantial modification. This removed a maximum fractional weight of 4×10^{-6} in the problems tested. Further research on eliminating this problem in the general case without approximation is ongoing.

5 Usage and Changes to Current Runs

As implemented, both the old and new algorithms are available simultaneously. The new α algorithm utilizes the current `ACODE` card with one modification. The value `KALPHA` can be set to 5, which enables the new tally estimator. Table 1 lists the options in the `ACODE` card.

Parameter	Description	Default
NSRCK	Number of source histories per generation	1000
RKK	Initial guess for k_{eff}	1.0
IKZ	Number of batches before α updates[a]	30
KCT	Total number of batches	IKZ + 100
MSRK	Number of source points for which storage will be allocated	Larger of 4500 or 2×NSRCK
KNRM	Normalization method for tallies: 0 = weight, 1 = histories	0
KALPHA	Estimation method for α: 1. Collision k for k–α iteration 2. Absorption k for k–α iteration 3. Track length k for k–α iteration 4. Differential operator perturbation 5. Tally-based α solve (NEW)	3
KALSAV	Number of batches before α is averaged and tallying begins	automatic [b]
KALREG	Batch to start ln-ln regression and reduce number of generations per batch from 4 to 2 (ignored if KALPHA = 5)	KALSAV + 2
MKRP	Maximum number of batches for which values are retained on MCTAL or RUNTPE files	6500
ALPHA	Initial guess for α, 10^8/s	0.0
ALMIN	Minimum value permitted for α, 10^8/s	0.0
KTARG	Target value for k_{eff}	1.0

Table 1: MCNP ACODE Parameters

[a] Other documents mention that this is the inactive cycles. This might lead to confusion in comparison to KCODE, in which tallies begin once inactive cycles end. For ACODE, tallies begin after KALSAV.

[b] The automatic feature of KALSAV checks to see that k_{eff} is converged to within one standard deviation of KTARG. It performs no convergence tests on the source. This will be tested in Sec. 6.4.

When KALPHA is set to 5, settling generations are not performed. In every batch, α is updated. As a consequence, KALREG is meaningless. In order to enable delayed neutrons, totnu must be explicitly added to the data cards. A prompt-only simulation will be performed if it is ignored.

A few changes had to be made to the current ACODE output. While geometries that ignore delayed neutrons typically have αs on the order of shakes in both the negative and positive directions, near critical geometries with delayed neutrons have much smaller αs. As a consequence, output in many locations has been replaced with scientific notation. In addition, Shannon entropy is now always output, as it is a useful tool for determining convergence.

6 Testing

For testing, several parameters were of interest. The first was the accuracy of the method. A simple deterministic benchmark was developed in order to do direct comparisons in Sec. 6.1. The second is performance. The figure of merit was compared between the k–α algorithm and the tally algorithm for delay-critical geometries and for prompt supercritical geometries in Sec. 6.2. The bias of the two algorithms were compared in Sec. 6.3. Then finally, a comparison of convergence is performed in Sec. 6.4.

For all tests that use real data, ENDF-B/VII.1 [5] was used at 293.6 K.

6.1 Infinite Medium Deterministic Comparison

One of the simplest options to ensure the accuracy of an algorithm is to create a deterministic benchmark. The complete approach used here is described in [8]. For this test, an infinite medium was filled with a material with cross-sections as listed in Table 2. All energy dependent functions (cross-sections, χ) were treated as constant from 10^{-11} to 20 MeV and zero otherwise. The term f_s is a scaling parameter to adjust the reactor from subcritical to prompt supercritical. In addition, the delayed precursor groups listed in Table 3 were used. Scattering was treated as isotropic.

Using this data, Eq. (4) was discretized into groups. This results in Eq. (10), where f is the scattering kernel and δ is the energy width of the

Parameter	Value
Σ_t	1.0 cm^{-1}
Σ_s	0.3 cm^{-1}
Σ_c	0.45 cm^{-1}
Σ_f	0.25 cm^{-1}
ν_p	$2.6 f_s$ cm^{-1}
ν_d	$0.2 f_s$ cm^{-1}
A	10
Speed of Light	2.997925×10^8 m/s
Mass of Neutron	939.58×10^6 MeV/c^2

Table 2: Neutronics Parameters for the α-Eigenvalue Benchmark

Group	λ, s^{-1}	p, Probability
1	0.0133	0.0350
2	0.0327	0.1807
3	0.1208	0.1725
4	0.3028	0.3868
5	0.8495	0.1586
6	2.8530	0.0664

Table 3: Delayed Precursor Parameters for the α-Eigenvalue Benchmark

Parameter	Value
Neutrons / Batch	100000
Batches	200
Batches Before α Updated	50
Batches Before Active	100
Initial α	0.0
Minimum α	$-1.33 \times 10^{-2} + 10^{-13}$ s^{-1}

Table 4: Infinite Medium Benchmark MCNP Simulation Parameters

group.

$$\begin{aligned} \frac{1}{\delta_j}\int_{E_j}^{E_{j+1}} dE \frac{1}{v(E)}\alpha\phi_j + \Sigma_t\phi_j &= \sum_i \chi_j\nu_p\Sigma_f\phi_i + \sum_i \chi_j\lambda_i C_i \\ &+ \sum_i \int_{E_j}^{E_{j+1}} dE' \int_{E_i}^{E_{i+1}} dE \frac{\Sigma_s f(\alpha_s E, E; E')}{\delta_i}\phi_i \\ \alpha C_j &= \sum_i p_j\nu_d\Sigma_f\phi_i - \lambda_j C_j \end{aligned} \tag{10}$$

The integrals over inverse velocity (relativistic) and scattering were performed numerically to high precision. Then Eq. (10) was converted into a matrix for which the eigenvalues can be directly solved. This was performed for values of f_s from 0.9 to 1.1.

This approach is only exact as the number of groups, N approaches infinity. As such, the number of groups was repeatedly doubled until at least the first ten digits were constant. In the case of $f_s = 1.1$, this was impractical, so sequence acceleration using Wynn's epsilon [15] was performed for $N = 2048, 4096, 8192, 16384, 32768$. All arithmetic with the exception of $f_s = 1.1$ used 256-bit reals.

These values were then compared to MCNP in which the exact same model was simulated using the tally α algorithm. These simulations were repeated 100 times with different random sequences in order to measure the true uncertainty. The simulation parameters are listed in Table 4.

The mean and the standard deviation of the mean are shown in Table 5 in comparison to the deterministic results. For the deterministic results, all stationary digits are listed. In general, the deterministic results were either

f_s	Deterministic α, s^{-1}	Mean α, s^{-1} 50 MCNP Runs	α Std. Dev of Mean 50 MCNP Runs	Det. Groups
0.9	$-1.297\,467\,136\,352\,483 \times 10^{-2}$	$-1.297\,467\,136\,352\,466 \times 10^{-2}$	2.5×10^{-18}	512
0.925	$-1.284\,043\,823\,409\,392 \times 10^{-2}$	$-1.284\,043\,823\,409\,359 \times 10^{-2}$	3.8×10^{-18}	512
0.95	$-1.254\,909\,434\,037\,18 \times 10^{-2}$	$-1.254\,909\,434\,037\,101 \times 10^{-2}$	9.6×10^{-18}	512
0.975	$-1.150\,917\,974\,504\,57 \times 10^{-2}$	$-1.150\,917\,974\,504\,125 \times 10^{-2}$	4.9×10^{-17}	512
1.0	$-6.385\,261\,566\,394 \times 10^{-14}$	-1.6960×10^{-16}	1.1×10^{-19}	512
1.025	$8.941\,945\,357\,03 \times 10^{-2}$	$8.941\,945\,357\,075\,2 \times 10^{-2}$	2.8×10^{-14}	512
1.05	$5.435\,470\,141\,3 \times 10^{-1}$	$5.435\,470\,141\,357\,7 \times 10^{-1}$	9.4×10^{-13}	512
1.075	$1.873\,125\,125 \times 10^{1}$	$1.873\,125\,126\,86 \times 10^{1}$	9.7×10^{-9}	2048
1.1	$4.560\,412\,921 \times 10^{7}$	$4.560\,53 \times 10^{7}$	2.1×10^{3}	seq. accel

Table 5: Results Comparing the Deterministic Solution to the Modified MCNP Solution

within statistics or within an absolute error of 10^{-12} s^{-1}. This gives a high degree of confidence in the implementation.

6.2 Analysis of Performance

In order to compare the performance of the old algorithm to the new one, a number of prompt α tests were run. The first block contains several critical assemblies from the ICSBEP Handbook [1]. In order to measure the performance of adding delayed neutron tallies, several prompt supercritical tests from [6] (problems 2, 3, and 4) were then tested in the second block.

Unless otherwise noted, all simulations used the parameters listed in Table 6. It is important to note that the old α eigenvalue calculation repeats batches in order to better converge α. As such, the old algorithm will run approximately twice the number of neutrons. This increase in run time is normalized out through the comparison of figure of merit. The figure of merit was calculated as:

$$\text{FOM} = \frac{1}{\sigma^2_{\text{true}} \bar{t}_{\text{total}}}$$

where $\bar{t}_{\text{total}}$ is the mean active simulation time. The standard deviation was computed by running each simulation 100 times with different random sequences. Each model was run using three of the original methods in MCNP

Parameter	Value
Neutrons / Batch	10000
Batches	600
Batches Before α Updated	20
Batches Before Active	100
Initial α	0.0
Minimum α	-10^8 s^{-1}

Table 6: Critical Assembly MCNP Simulation Parameters

Benchmark	Case	Name	k_{eff}
ieu-met-fast-007	—	BIG TEN	1.00461 ± 0.00003
u233-met-fast-006	—	Flattop-23	0.99879 ± 0.00004
heu-met-fast-028	—	Flattop-25	1.00282 ± 0.00004
pu-met-fast-006	—	Flattop-Pu	0.99999 ± 0.00005
heu-met-fast-001	—	Godiva	0.99979 ± 0.00004
pu-met-fast-001	—	Jezebel	0.99992 ± 0.00004
u233-met-fast-001	—	Jezebel-233	0.99985 ± 0.00004
pu-met-fast-008	2	THOR	0.99771 ± 0.00004
heu-met-inter-006	1	Zeus-1	0.99299 ± 0.00005

Table 7: Critical Assembly Parameters

(k_{c}, k_{tr}, k_{a}) and all figures of merit are normalized to the best result of the three.

6.2.1 Critical Assemblies

The nine critical assemblies tested are listed in Table 7. These models were chosen as a roughly representative spread of different materials and geometries. The geometries span from a bare spheres (Godiva, Jezebel) to large reflected cylinders (Zeus-1, BIG TEN). The k_{eff} was calculated using 500 thousand neutrons per batch.

Due to errors in the nuclear data and the experiment, these values of k diverge from the experimental results. For the rest of these tests, these simulations were treated as delay critical by setting the `KTARG` card to the

Model	k–α α, s^{-1}	Prompt Tally α, s^{-1}
BIG TEN	$-1.1736 \pm 0.0042 \times 10^5$	$-1.1615 \pm 0.0052 \times 10^5$
Flattop-23	$-3.147 \pm 0.027 \times 10^5$	$-2.847 \pm 0.026 \times 10^5$
Flattop-25	$-3.982 \pm 0.017 \times 10^5$	$-3.861 \pm 0.017 \times 10^5$
Flattop-Pu	$-2.311 \pm 0.027 \times 10^5$	$-2.058 \pm 0.027 \times 10^5$
Godiva	$-1.1503 \pm 0.0053 \times 10^6$	$-1.1432 \pm 0.0052 \times 10^6$
Jezebel	$-7.061 \pm 0.092 \times 10^5$	$-6.72 \pm 0.10 \times 10^5$
Jezebel-233	$-1.0790 \pm 0.0096 \times 10^6$	$-1.064 \pm 0.011 \times 10^6$
THOR	$-2.417 \pm 0.028 \times 10^5$	$-2.075 \pm 0.027 \times 10^5$
Zeus-1	$-3.752 \pm 0.016 \times 10^3$	$-3.433 \pm 0.016 \times 10^3$

Table 8: Comparison of Prompt α Eigenvalues, Critical Assemblies

k given above. It is important to note that this is a rough approximation. Some of the reactor k-eigenvalues are not precisely 1, and scaling just k does not correct for all of the systematic errors present.

The computed value of prompt α is listed in Table 8. Values shown are for the mean and standard deviation of the mean of the estimator with the smallest standard deviation. It is notable that the new algorithm yields answers that are different from the old one by many standard deviations. The most extreme case, Zeus-1, has a discrepancy of nearly 14 standard deviations. As will be shown in Sec. 6.3, these discrepancies are actually due in large part to bias effects from too few neutrons per batch.

As for the performance, the relative figure of merit is tabulated in Table 9. In all tests, the combined α direct tally estimator outperformed the stock algorithm, with improvements from 15 to 90% in figure of merit. The combined estimator often performed as well as the best tally estimator, but usually not much better. The track length tally α tended to outperform the collision tally except in the case of BIG TEN.

The last test of interest is to quantify the impact of the current MCNP approach of running multiple generations per batch but only tallying and updating α on the last one. As this is expected to most benefit reactors with large dominance ratios, BIG TEN was rerun with two generations per batch with the tally α algorithm. The resulting α and FOM are tabulated in Table 10. In this particular case, running multiple generations per batch to converge the source in general reduces the figure of merit. It does not reduce it by a factor of two, as would be expected due to the runs taking twice as

Model	k–α FOM			Prompt Tally FOM		
	Col.	Abs.	Tr. Ln.	Col.	Tr. Ln.	Combined
BIG TEN	0.98	0.99	1.00	1.15	1.10	1.15
Flattop-23	0.73	0.75	1.00	1.81	1.91	1.90
Flattop-25	0.97	0.79	1.00	1.76	1.82	1.82
Flattop-Pu	0.85	1.00	0.88	1.63	1.74	1.73
Godiva	0.69	0.72	1.00	1.25	1.79	1.89
Jezebel	0.38	0.41	1.00	1.01	1.49	1.49
Jezebel-233	0.44	0.43	1.00	0.84	1.39	1.48
THOR	0.49	0.61	1.00	1.81	1.87	1.88
Zeus-1	1.00	0.97	0.87	1.78	1.84	1.88

Table 9: Normalized Figures of Merit, Critical Assemblies

Mode	α, s^{-1}	Normalized FOM
k–α	$-1.1736 \pm 0.0042 \times 10^5$	1.00
Tally, One Gen.	$-1.1615 \pm 0.0052 \times 10^5$	1.15
Tally, Two Gen.	$-1.1654 \pm 0.0043 \times 10^5$	0.83

Table 10: BIG TEN, Multiple Generations per Batch

long. As such, the per-batch variance is decreased, but the cost of the batch is increased more than the decrease in the variance.

6.2.2 Supercritical Assemblies

For the supercritical test, the three prompt supercritical problems from [6] were run. Problem 2 corresponds to a double density Godiva sphere. Problem 3 corresponds to a double density Godiva reflected with 30 cm of water. Problem 4 corresponds to a Godiva sphere diluted 100:1 water to uranium. All three geometries have k-eigenvalues well in excess of prompt supercritical. As such, the results should be the same with or without delayed neutrons. This allows for direct comparison of the performance of the new algorithm with delayed neutrons as compared to the stock algorithm.

Table 11 lists the mean α-eigenvalues for all three test problems. All values are effectively identical within statistics. Table 12 lists the relative

Model	k–α α, s^{-1}	Prompt Tally α, s^{-1}	Delay Tally α, s^{-1}
Problem 2	$1.445\,85 \pm 0.000\,11 \times 10^8$	$1.446\,10 \pm 0.000\,11 \times 10^8$	$1.446\,04 \pm 0.000\,12 \times 10^8$
Problem 3	$1.464\,79 \pm 0.000\,11 \times 10^8$	$1.465\,04 \pm 0.000\,12 \times 10^8$	$1.464\,92 \pm 0.000\,11 \times 10^8$
Problem 4	$6.708\,86 \pm 0.000\,44 \times 10^5$	$6.709\,43 \pm 0.000\,39 \times 10^5$	$6.708\,73 \pm 0.000\,36 \times 10^5$

Table 11: Comparison of α Eigenvalues, Supercritical Assemblies

Model	Prompt Tally FOM			Delay Tally FOM		
	Col.	Tr. Ln.	Combined	Col.	Tr. Ln.	Combined
Problem 2	1.86	1.60	1.74	1.02	1.15	1.12
Problem 3	1.66	1.55	1.67	1.38	1.30	1.37
Problem 4	2.26	2.21	2.26	1.78	1.77	1.77

Table 12: Normalized Figures of Merit, Supercritical Assemblies

figure of merit as compared to the best k–α estimator. In this case, the tally algorithm significantly outperformed the k–α algorithm even with the added cost of tallying delayed particles. There is indeed a slowdown corresponding to the time required to calculate the α scaled ν values and the time required to tally the delayed neutron precursor populations. That slowdown is on the order of 25-35%.

6.3 Analysis of Bias

As mentioned in the theory sections, one of the drawbacks of the new tally algorithm is that the nonlinear transform used to generate α will induce bias. However, it is well-known that k is biased [4], so it is also anticipated that the k–α algorithm will also be biased. The important question then is: how does the bias compare between algorithms?

To test this, the Zeus-1 model, which had the largest discrepancy in Sec. 6.2.1, was run with a fixed 500 million active neutrons. The number of batches and particles per batch were adjusted to examine how the α-eigenvalue converges. In Fig. 1, the final α-eigenvalue is plotted as a function of neutrons per batch. Both algorithms are indeed converging to the same value of roughly -3.5×10^3 s^{-1} (within statistics). However, while the tally-based α starts relatively close to the final answer, the k–α solve is heavily

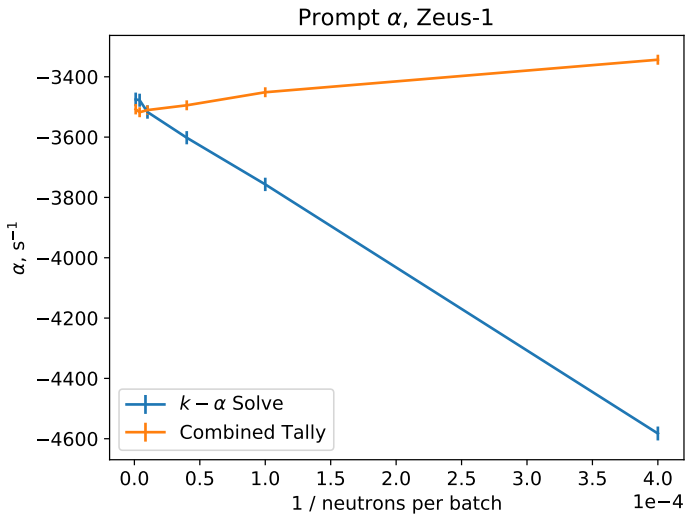

Figure 1: Convergence of Prompt α with Neutrons / Batch

biased towards negative α. This bias explains the discrepancy in Sec. 6.2.1.

With this understanding, the critical assemblies were rerun with 2 million neutrons per batch. The results are shown in Table 13. Here, due to the cost of these simulations, the error is the estimate as reported by the simulation and not the true standard deviation. This value is liable to be an underestimation. Under these circumstances, all experiments match within two standard deviations.

From the previous section on performance, the FOM benefit of the tally algorithm usually did not exceed 2. Since the tally α and k–α iteration were run with identical settings and k–α typically runs two generations per batch, k–α takes roughly twice as long to run. As such, it is anticipated that the k–α solution in Table 13 is slightly more accurate despite having worse estimated standard deviations. Using this value as the reference, the bias in the 10 thousand neutrons per batch solutions from Table 8 can be estimated. This is tabulated in Table 14. Here, for all geometries, the tally estimator had lower bias. In addition, the tally estimator is effectively within statistics on all values with the possible exception of Zeus-1.

The convergence of the results with delayed neutrons enabled was also analyzed. The Zeus-1 model, in which `KTARG` is set to k_{eff} and delayed neutrons are enabled, should ideally yield an α of zero. As shown in Fig. 2,

Model	k–α α, s^{-1}	Prompt Tally α, s^{-1}
BIG TEN	$-1.1559 \pm 0.0028 \times 10^5$	$-1.1577 \pm 0.0018 \times 10^5$
Flattop-23	$-2.840 \pm 0.014 \times 10^5$	$-2.8704 \pm 0.0058 \times 10^5$
Flattop-25	$-3.874 \pm 0.011 \times 10^5$	$-3.8680 \pm 0.0051 \times 10^5$
Flattop-Pu	$-2.065 \pm 0.016 \times 10^5$	$-2.0813 \pm 0.0051 \times 10^5$
Godiva	$-1.1328 \pm 0.0035 \times 10^6$	$-1.1374 \pm 0.0028 \times 10^5$
Jezebel	$-6.561 \pm 0.064 \times 10^5$	$-6.555 \pm 0.051 \times 10^5$
Jezebel-233	$-1.0585 \pm 0.0070 \times 10^6$	$-1.0619 \pm 0.0059 \times 10^6$
THOR	$-2.114 \pm 0.020 \times 10^5$	$-2.0845 \pm 0.0062 \times 10^5$
Zeus-1	$-3.482 \pm 0.011 \times 10^3$	$-3.500 \pm 0.013 \times 10^3$

Table 13: Prompt α Eigenvalues, Critical Assemblies, 2 Million N / Batch

Model	k–α Bias		Tally Bias	
	Percent	Std. Deviations	Percent	Std. Deviations
BIG TEN	1.5 %	3.5	0.48 %	0.95
Flattop-23	10.8 %	10.1	0.24 %	0.23
Flattop-25	2.8 %	5.3	0.32 %	0.61
Flattop-Pu	11.9 %	7.9	0.36 %	0.24
Godiva	1.5 %	2.8	0.92 %	1.7
Jezebel	7.6 %	4.5	2.4 %	1.3
Jezebel-233	1.9 %	1.7	0.52 %	0.43
THOR	14.3 %	8.8	1.8 %	1.2
Zeus-1	7.8 %	13.7	1.4 %	2.5

Table 14: Bias of the 10 Thousand Neutrons per Batch Simulations

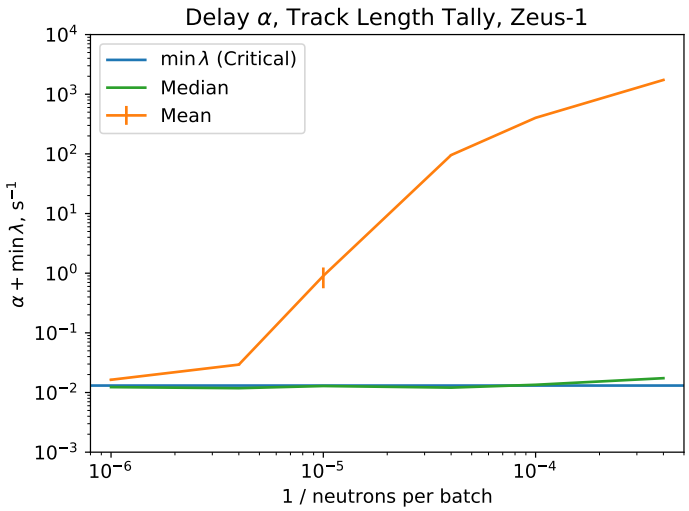

Figure 2: Convergence of Delayed α with Neutrons / Batch

it is quite challenging for a simulation to yield an accurate value. At 2500 neutrons per batch, the mean indicates that the reactor is effectively prompt supercritical. As the neutrons per batch is increased, the mean begins to converge towards critical. The answer is still unreasonably inaccurate until at least 250 thousand. At 1 million, the mean and median still diverge by more than 3 standard deviations. It is also notable that the median is far less sensitive to the neutrons per batch. Even at 2500, the median reports a near-critical value.

In order to better grasp the situation, the distributions of α were plotted for the 2500, 100 thousand, and 1 million neutrons per batch runs. These results are shown in Fig. 3. In the 2500 neutron case, the distribution is fully multimodal. The rightmost mode corresponds to a prompt supercritical tally. The remaining modes indicate the general structure of Eq. (7). Increasing to 100 thousand neutrons per batch, the distribution becomes unimodal. However, there are rare tallies of very large values of α, skewing the distribution. In this particular case, only 1.9% of the batches exceed the mean. Further increasing to 1 million results in a single mode without significant outliers.

This last distribution was plotted in a quartile-quartile plot to compare against both a normal and a log-normal distribution in Fig. 4. In the log-normal case, $\min \lambda_i$ was added to α to ensure positivity. It is clear that the

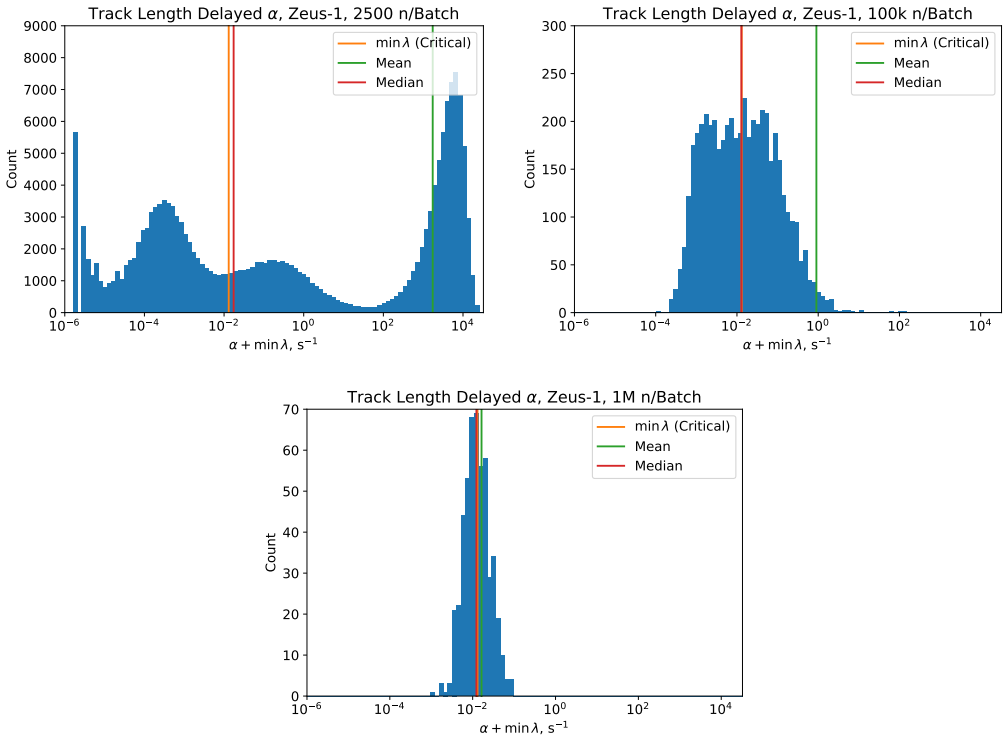

Figure 3: α_{tr} density as a function of batches.

distribution deviates substantially from both normal and log-normal, even with 1 million neutrons per batch.

6.4 Convergence

For this final test, the convergence of the algorithms was investigated. Before a user performs tallies, it is vital that all inputs to the batch must be stationary. In the α-eigenvalue, this includes both α and the source distribution. The source convergence is nominally quantified by Shannon entropy.

In order to test convergence, a problem with poor convergence properties is needed. As such, a Godiva sphere was expanded in radius to 2 meters and then reflected with 30 cm of water. The large mass should in general have a high dominance ratio. This will reduce the convergence rate.

Both the k–α and tally α algorithms were used to simulate this sphere. 1 million particles per batch were used and 2000 batches recorded. α was allowed to vary starting at batch 20. In this particular case, the k–α method ran 4 generations per batch for batches 21-800. The results are plotted in Fig. 5. Due to the multiple generations per batch, the k–α method converged much quicker on a batch-scale. When renormalized for the same number of generations (and, thus, simulation time), performance is effectively identical.

The value of α was also plotted similarly in Fig. 6. Here, α was always effectively converged by batch 50, so further points are not plotted. One interesting thing to note is that the tally α rises smoothly to the final value, but the k–α method has a ringing phenomenon. Overall, the Shannon entropy converges long after the eigenvalue, as is the case in k-eigenvalue simulations.

It is also worth testing when one should switch from k convergence to α convergence. In the previous tests, generation 20 was used as the transition. Using the tally algorithm, this test was repeated with a cutoff of 600 (when k Shannon entropy became stationary), and a cutoff of 1100. The results are shown in Fig. 7, along with the k-eigenvalue plot. For the 20 and 600 case, both simulations effectively had the same convergence in the end, with stationarity near batch 900. As would be expected, the 1100 case was not stationary until batch 1400. At least for this problem, there is a disadvantage to setting `IKZ` large and no apparent disadvantage to setting `IKZ` small.

The automatic feature of `KALSAV` was also used for the 600 and 1100 cases. k was within 1 standard deviation by batch 609 and 1107 respectively. Since the source was far from converged at both points, any tallies would likely be incorrect.

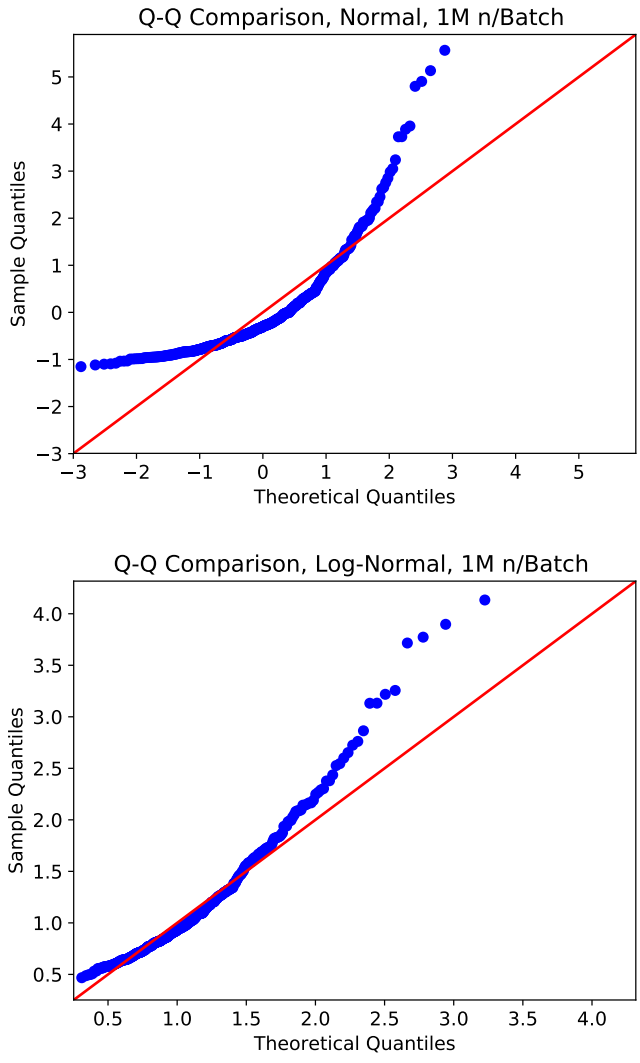

Figure 4: Q-Q Plots vs. Normal and Log-Normal Distributions for α.

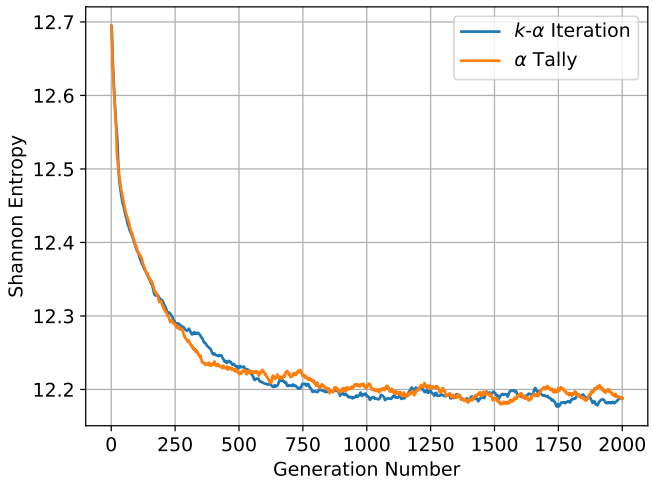

Figure 5: Convergence of Shannon Entropy

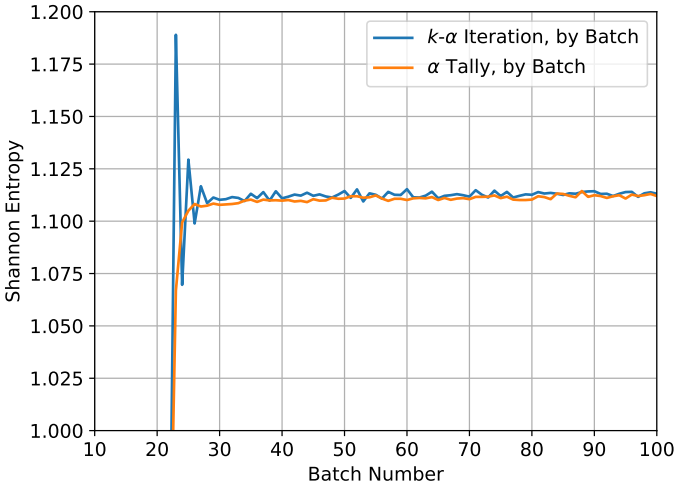

Figure 6: Convergence of α

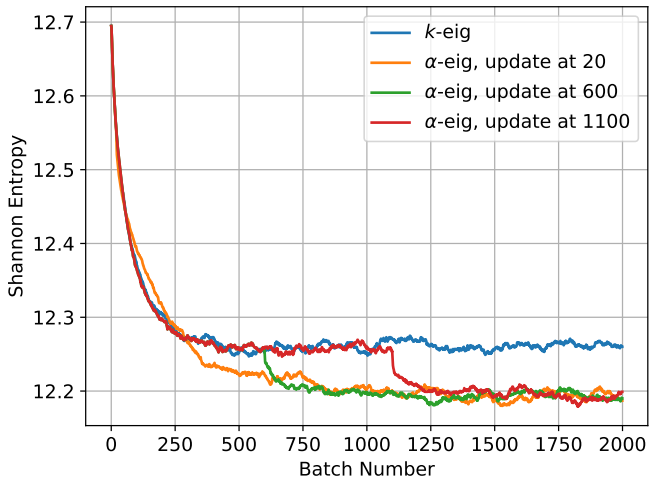

Figure 7: Shannon entropy with different α start times

7 Recommendations for Use

Based on the prior sections, a few recommendations can be made for the correct use of `ACODE` with the new modifications. Recommendations for each option of the `ACODE` card are listed below.

`NSRCK` - Number of source histories per batch

As demonstrated in Sec. 6.3, the number of source particles should be large enough such that the bias of the α distribution is acceptable. One quick test is to compare the mean and the median. One could also examine the skew and the excess kurtosis of the distribution.

There are two cases in which one would need more neutrons per batch than commonly anticipated. The first case involves any simulation near critical. For these, the relative magnitude of the bias as compared to α can grow quite large. In extreme cases, this can prevent knowing if a geometry is subcritical or supercritical.

The second case involves the prompt critical boundary in delay neutron simulations. Any simulation in which α can be on both sides of this boundary will have α heavily biased positive. For the test of delay critical Zeus-1, there were still indications of bias at 1 million particles per batch. Even more

neutrons will be required as one approaches prompt supercritical.

`RKK` - Initial guess for $k_{\mathbf{eff}}$

This scaling parameter is only needed if reactor is initially very distant from critical. A large value will reduce the number of particles written to the fission bank in the first batch, possibly preventing crashes. Later batches will use the calculated k value and not `RKK`.

`IKZ` - Number of batches before α updates

A sufficient amount of time is needed between `IKZ` and `KALSAV` for both α and the source distribution to stabilize. There appears to be nearly no reason to set `IKZ` to a large value. As such, a value of 5 to 50 is adequate.

`KCT` - Total number of batches

The value `KCT` - `KALSAV` determines the number of active batches. This needs to be large enough that the distribution of batches can approach Gaussian. As such, this should be at least 50 larger than `KALSAV`.

`MSRK` - Number of source points for which storage will be allocated

When simulating a model with a large negative α, increasing this value might improve stability and prevent source bank overruns. However, there are some simulations in which the time source can grow without bounds, and no value of `MSRK` will prevent crashes. This is currently an open problem.

`KNRM` - Normalization method for tallies

This setting should be left as default.

`KALPHA` - Estimation method for α

The new tally algorithm outperforms the old one on all problems tested, so it is suggested to enable the new algorithm via `KALPHA` = 5.

KALSAV - Number of cycles before α is averaged and tallying begins

This should be large enough that both α and the source are converged. It is recommended to run the simulation with a smaller NSRCK and observe the Shannon entropy. If the Shannon entropy appears converged at iteration N, set KALSAV a bit higher than N. Upon completion of the new simulation, examine the Shannon entropy again to determine if KALSAV was sufficiently large. This should not be set to automatic, as it only checks for k convergence and will ignore source convergence.

KALREG - Batch to reduce internal settling generations and start ln-ln regression

Leave as default.

MKRP - Number of batches for which data is retained

In order to use some of the diagnostics at the end of the simulation, such as the median, MKRP must be larger than KCT. This allows all of the α values to be available at once.

ALPHA - Initial guess

Setting ALPHA to the expected value of α might improve convergence. However, it is not necessary.

ALMIN - Minimum α

Some capabilities are incompatible with negative α, such as DXTRAN spheres. In addition, analog capture may give zero weights with α negative. If this is a concern, ALMIN should be set to zero. Otherwise, there are no ill effects to setting ALMIN too low. Conversely, if ALMIN is too large (such that a batch α needs resetting), it might bias the answer. For delayed neutron simulations, the batch α will never go below $-\min\lambda$, so any ALMIN below that value will have identical results.

KTARG - Target $k_{\mathbf{eff}}$

If the k_{eff} of the geometry is known to be a certain value, but the computed k_{eff} is different, this can be used to adjust the problem closer to the right

answer. An example is described in Sec. 6.2.1. It is important to note that this is a coarse approximation, as the discrepancy is almost certainly not due to a proportionality error in the fission data.

8 Conclusions

These modifications to MCNP in general improve performance and expand capabilities. The new tally α-eigenmode calculator was found to have a 15-126% higher figure of merit as compared to the best result from stock k–α iteration. The addition of support for delayed neutrons during α simulation allows for analysis of a wide variety of new problems. Comparisons against a deterministic benchmark show solutions that are either within statistics or within 10^{-12} s^{-1} absolute error.

In addition, an analysis of convergence of α for both the k–α and tally algorithms was performed. One interesting finding concerns the bias of the final α. In several cases, it was found that 10 thousand neutrons per batch was insufficient. When the prompt α of Zeus-1 was calculated, errors in α on the order of 10% occurred at when using the k–α iteration. The tally algorithm demonstrated a decreased bias at the same point.

An even more extreme case of bias occured when the Zeus-1 model was simulated with delayed neutrons. The effect of the prompt supercritical values of α biased results positive. α only approached the correct near-critical value with 250 thousand neutrons per batch, and there were indications that even the 1 million neutrons per batch solution was biased. For this reason, it is recommended that such simulations are performed with care.

9 Acknowledgements

This work was supported by the US-DOE-NNSA programs for Nuclear Criticality Safety and Advanced Simulation & Computing.

References

[1] International handbook of evaluated criticality safety benchmark experiments. Technical Report NEA/NSC/DOC(95)03 (2009 Edition), OECD Nuclear Energy Agency, 2009.

[2] George I Bell and Samuel Glasstone. Nuclear reactor theory. Technical report, US Atomic Energy Commission, Washington, DC (United States), 1970.

[3] D Brockway, P Soran, and P Whalen. Monte-Carlo eigenvalue calculation. In *Monte-Carlo Methods and Applications in Neutronics, Photonics and Statistical Physics*, pages 378–387. Springer, 1985.

[4] Forrest B Brown. A review of Monte Carlo criticality calculations–convergence, bias, statistics. *Proceedings M&C*, pages 3–7, 2009.

[5] MB Chadwick, M Herman, P Obložinskỳ, Michael E Dunn, Y Danon, AC Kahler, Donald L Smith, B Pritychenko, Goran Arbanas, R Arcilla, et al. ENDF/B-VII.1 nuclear data for science and technology: cross sections, covariances, fission product yields and decay data. *Nuclear data sheets*, 112(12):2887–2996, 2011.

[6] Dermott E Cullen, Christopher J Clouse, Richard Procassini, and Robert C Little. Static and dynamic criticality: are they different? Technical report, Lawrence Livermore National Lab., Livermore, CA (US), 2003.

[7] Theodorus Jozef Dekker. A floating-point technique for extending the available precision. *Numerische Mathematik*, 18(3):224–242, 1971.

[8] Colin Josey and Forrest B. Brown. A new Monte Carlo alpha-eigenvalue estimator with delayed neutrons. In *Proceedings of the American Nuclear Society (2018 ANS Annual Meeting)*. ANS, 2018, Accepted.

[9] Davide Mancusi and Andrea Zoia. Chaos in eigenvalue search methods. *Annals of Nuclear Energy*, 112:354–363, 2018.

[10] Philippe Pébay. Formulas for robust, one-pass parallel computation of covariances and arbitrary-order statistical moments. Technical report, Sandia National Laboratories, 2008.

[11] Philippe Pébay, Timothy B. Terriberry, Hemanth Kolla, and Janine Bennett. Numerically stable, scalable formulas for parallel and online computation of higher-order multivariate central moments with arbitrary weights. *Computational Statistics*, 31(4):1305–1325, Dec 2016.

[12] Hyung Jin Shim, Sang Hoon Jang, and Soo Min Kang. Monte Carlo alpha iteration algorithm for a subcritical system analysis. *Science and Technology of Nuclear Installations*, 2015, 2015.

[13] Todd J Urbatsch, R Arthur Forster, Richard E Prael, and Richard J Beckman. Estimation and interpretation of k_{eff} confidence intervals in MCNP. *Nuclear technology*, 111(2):169–182, 1995.

[14] BP Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420, 1962.

[15] Peter Wynn. On a device for computing the $e_m(s_n)$ transformation. *Mathematical Tables and Other Aids to Computation*, pages 91–96, 1956.

[16] Tao Ye, Chaobin Chen, Weili Sun, Benai Zhang, and Dongfeng Tian. Prompt time constants of a reflected reactor. Technical report, 2008.